Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A109 047

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
On the Power of Probabilistic Choice in Synchronous Parallel Computations

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
TR-30-81

7. AUTHOR(s)
John H. Reif

2

8. CONTRACT OR GRANT NUMBER(s)
N00014-80-C-0674,
[illegible]

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Harvard University
Cambridge, MA 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

12. REPORT DATE
November, 1981

13. NUMBER OF PAGES
27

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
same as above

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

16. DISTRIBUTION STATEMENT (of this Report)
unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Probabilistic algorithm, randomized algorithm, nonuniform, parallel algorithm, parallel speedup, perfect matching, matrix multiplication, polynomial irreducibility, computational complexity.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
see Reverse

20.

SUMMARY

This paper introduces probabilistic choice to synchronous parallel machine models; in particular parallel RAMs. The power of probabilistic choice in parallel computations is illustrated by $O(\log n)$ time algorithms for connectivity and recognizing bipartite graphs and $O(\log n)^2$ time algorithms for testing if a graph has a perfect matching, testing in time $O(n)$ irreducibility of polynomials over finite fields. We characterize the computational complexity of time, space, and processor bounded probabilistic parallel RAMs in terms of the computational complexity of probabilistic sequential RAMs. We show that parallelism uniformly speeds up time bounded probabilistic, sequential RAM computations by nearly a quadratic factor. We also show that probabilistic choice can be eliminated from parallel computations by introducing nonuniformity.

ON THE POWER OF PROBABILISTIC CHOICE IN SYNCHRONOUS PARALLEL COMPUTATIONS

John H. Reif

TR-30-81

ON THE POWER OF PROBABILISTIC CHOICE
IN SYNCHRONOUS PARALLEL COMPUTATIONS

by

John H. Reif

Aiken Computation Laboratory
Division of Applied Science
Harvard University
Cambridge, Mass.

November, 1981

*This work was supported in part by the National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0674.

(Squared)

SUMMARY

This paper introduces probabilistic choice to synchronous parallel machine models; in particular parallel RAMs. The power of probabilistic choice in parallel computations is illustrated by $O(\log n)$ time algorithms for connectivity and recognizing bipartite graphs and $O(\log n)^2$ time algorithms for testing if a graph has a perfect matching, testing in time $O(n)$ irreducibility of polynomials over finite fields. We characterize the computational complexity of time, space, and processor bounded probabilistic parallel RAMs in terms of the computational complexity of probabilistic sequential RAMs. We show that parallelism uniformly speeds up time bounded probabilistic, sequential RAM computations by nearly a quadratic factor. We also show that probabilistic choice can be eliminated from parallel computations by introducing nonuniformity.

1. INTRODUCTION

Probabilistic choice is the use of randomly chosen moves in an otherwise deterministic computation given a fixed input. The introduction of probabilistic choice in *sequential computations* leads to considerable improvement to the computational complexity of various number theoretic problems [Berlekamp, 70], [Rabin, 74], [Solovay and Strassen, 77], [Adleman, Manders, and Miller, 79], [Rabin, 80], [Zippel, 79] to combinatorial problems on graphs and matroids [Lovász, 80], to testing polynomial identities [Schwartz, 80], and testing program equivalence [Ibarra and Moran, 80].

Recently, [Rabin, 80], [Lehman and Rabin, 80],[Francez and Rodeh, 80], [Reif and Spirakis, 81 and 82] have utilized probabilistic choice in *synchronization algorithms* for asynchronous multiprocesses systems.

This paper investigates the use of probabilistic choice in *synchronous parallel machines*. We present a pair of simulation results (Theorems 4.1 and 4.2) which relate probabilistic sequential and probabilistic parallel computations on RAMs. By parallel simulation of previously known probabilistic sequential algorithms [Aleliunas, et al., 79], our Theorem 4.1 immediately yields as corollaries the fastest known parallel algorithms for a variety of combinatorial problems such as an $O(\log n)$ time test if there exists a path between two vertices of a undirected graph and an $O(\log n)$ time test if graph is bipartite. Both these probabilistic parallel algorithms use $O(n^3 \log n)$ processors. Previously the fastest known parallel algorithm for these problems required $O(\log^2 n)$ [Csanky, 76].

We give $O(\log n)^2$ time probabilistic P-RAM algorithms for testing if a graph of n vertices has a perfect matching, and an $O(n)$ time test if a polynomial of degree $O(n)$ has a root over $GF(p^n)$. (Also, recently [Reischuk, 81] has shown that a probabilistic parallel RAM can sort in time $O(\log n)$ with $O(n)$ processors.)

We have an interesting theoretical result (Theorem 5) for speeding up a log-cost (unit-cost, respectively) probabilistic sequential RAM computation of time $T(n)$, by simulation on a probabilistic parallel RAM in log-cost time $O(T(n)^{1/2} \log T(n))$ (in unit-cost time $O(T(n)(\log T(n)) \log(T(n)I(n)))^{1/2}$, respectively, where $I(n)$ is the maximum integer operated upon the simulated unit-cost probabilistic RAM). Previously, [Dymond, 80] proved a quadratic speedup of deterministic multitape Turing machines; however he considered the simulation of neither probabilistic machines nor RAMs.

[Adleman, 78] has previously proved that probabilistic choice can be eliminated in sequential computations if there is no error of acceptance. Theorem 6 of Section 6 proves that probabilistic choice can be eliminated from probabilistic parallel RAMs with both errors of acceptance and errors of rejection by introducing nonuniformity, with some increase of time and processor bounds which may be traded off. This implies there exists non-uniform deterministic parallel RAMs which can in unit-cost time $O(\log n)$ test if a graph of n vertices is connected, and in time $O(\log n)^2$ test if a graph of n vertices has a perfect matching, and in time $O(n)$ test if a polynomial of degree $O(n)$ has a root in $GF(p^n)$.

2. DEFINITIONS OF PROBABILISTIC MACHINES

2.1 Abstract Machine Types

Before describing our probabilistic parallel machines, it is useful to define probabilistic (and also deterministic and nondeterministic) machine types abstractly, without reference to the particular details of operation of the machines.

Let M be a fixed machine. A *configuration* of M is a finite string I over a fixed finite alphabet describing the current state and storage contents of M. Let $\mathcal{I}$ be the set of configurations of M. Let $\mathcal{I}_A \subseteq \mathcal{I}$ be the set of *accepting configurations* of M. Let Σ be the finite input alphabet of M. Given an input string $\omega \in \Sigma^*$, let $I_0(\omega) \in \mathcal{I}$ be the corresponding *initial configuration* of M. Let $\vdash \subseteq \mathcal{I} \times \mathcal{I}$ be the *next move relation* for M; for each $I \in \mathcal{I}$, $\mathrm{NEXT}(I) = \{I' | I \vdash I'\}$ is the set of possible configurations derived from I by a single move of M. (We assume there is no next move from an accepting configuration.) In a *nondeterministic* machine, any $I' \in \mathrm{NEXT}(I)$ may be chosen nondeterministically. In a *probabilistic machine*, each $I' \in \mathrm{NEXT}(I)$ is chosen with equal probability, independently of previous and succeeding choices. In a *deterministic machine* M, $|\mathrm{NEXT}(I)| \leq 1$ for all $I \in \mathcal{I}$.

Given a fixed input string $\omega \in \Sigma^*$, a *computation sequence* of M is a maximal length sequence of configurations $I_0, I_1, \ldots$ such that $I_0 = I_0(\omega)$ and $I_{i-1} \vdash I_i$ for $i = 1, 2, \ldots$. The computation sequence is *accepting* if it is finite and the last configuration is accepting. In a deterministic or nondeterministic machine, M accepts ω iff there exists an accepting computation sequence from $I_0(\omega)$. In a probabilistic machine, M *accepts* ω

iff Prob(COMP(ω) is accepting) $\geq 1/2$, where COMP(ω) is a random computation sequence from $I_0(\omega)$ (generated by random next moves as defined above). Let the *language accepted* by M be $L(M) = \{\omega \in \Sigma^* | M$ accepts $\omega\}$.

2.2 Error Restricted Probabilistic Machines

Let M be a probabilistic machine which accepts language L(M). Let the *acceptance error* $\varepsilon_A(n)$ and the *rejection error* $\varepsilon_R(n)$ be the minimum functions such that for all $n \geq 0$, $\omega \in \Sigma^n$,

(i) if $\omega \notin L(M)$ then

Prob{COMP(ω) is accepting} $\leq \varepsilon_A(n)$

(ii) If $\omega \in L(M)$ then

Prob{COMP(ω) is not accepting} $\leq \varepsilon_R(n)$.

Note that by definition $\varepsilon_A(n) \leq 1/2$ and $\varepsilon_R(n) \leq 1/2$.

For deterministic or nondeterministic machines M, M' let $M \approx M'$ if $L(M) = L(M')$. For two probabilistic machines M, M', let $M \approx M'$ have both the same error of acceptance and the same error of rejection.

Let M be a BP-*probabilistic machine* if there exists a constant $\varepsilon < 1/2$ such that for all $n \geq 0$, $\varepsilon \geq \max(\varepsilon_A(n), \varepsilon_R(n))$. Thus a BP-probabilistic machine has a constant upper bound, which is less than 1/2, on errors of acceptance and rejection.

Let M be a R-*probabilistic machine* if there exists a constant $\varepsilon < 1/2$ such that for all $n > 0$, $\varepsilon \geq \varepsilon_R(n)$, and M never has an accepting computation on any input string $\omega \in \Sigma^* - L(M)$.

2.3 Probabilistic Sequential Machines

A nondeterministic Turing machine may be made a *probabilistic Turing machine* by allowing next moves to be chosen randomly with equal probability, as described in Sec. 2.1. See [Simon, 75] for a discussion of probabilistic Turing machines with unrestricted errors and see [Adleman, 78] for some results for R-probabilistic Turing machines. [Bennett and Gill, 81] discuss these and various other classes of probabilistic Turing machines.

Our principal sequential machine model is the probabilistic Random Access Machine (RAM), which is defined here similarly to [Aho, Hopcroft and Ullman, 74], except we allow the RAM probabilistic choice. A *probabilistic* RAM consists of

(1) an infinite sequence of memory locations $m_0, m_1, \ldots$ each of which are indexed by and contain a nonnegative integer

(2) a fixed set of registers R each of which contains a nonnegative integer

(3) a probabilistic finite state control which allows the following operations:

(a) for any registers $r_1, r_2 \in R$, *load* (or *read*) the contents of r_1 into (or from, respectively) the contents of global memory location m_i, where i is the current contents of register r_2.

(b) for any registers $r_1, r_2, r_3 \in R$, apply an addition, subtraction, multiplication, or division operation on the contents of registers r_1, r_2 and load the result into register r_3.

(Note: we round noninteger rationals to the next lower integer. Also, we substitute 0 for the result of a subtraction which is negative.)

A *unit cost* RAM is charged 1 step for each of the above operations; a *log-cost* RAM is charged $\lceil \log(x+2) \rceil$ steps for each of the above operations which are on integers of size x.

We assume a binary input alphabet $\{0,1\}$. Given an input string $\omega \in \{0,1\}^*$, each memory location m_{i-1} initially contains the i-th bit of ω for $1 \leq i \leq |\omega|$, m_n contains 2, and all other memory locations and registers are initially 0. The memory location $m_0,\ldots,m_n$ are read-only, and cannot be loaded into. Also, we assume the finite control has distinguished *initial* and *accepting* states. A configuration is accepting if the machine is in the accepting state. The probabilistic RAM *accepts* input ω if with probability $> 1/2$ a random computation sequence is accepting. The probabilistic RAM has *time bound* $T(n)$ (*space bound* $S(n)$, *integer bound* $I(n)$) if on all inputs of length n and accepting computation sequences, the machine takes $\leq T(n)$ steps (uses $\leq S(n)$ space, operates on integers $\leq I(n)$, respectively). Note that we have defined steps differently for unit-cost and log-cost RAMs. Furthermore, a log-cost RAM (unit-cost RAM, respectively) is charged $\log(x+2)$ (1, respectively) units of space for each noninput memory location and register utilized in an accepting computation, where x is the largest integer stored in that memory location or register.

2.4 Probabilistic Parallel RAMs

Our principle parallel machine model is the Parallel Random Access Machine (P-RAM), similar to that defined in [Fortune and Wyllie, 78] and [Wyllie, 79]. However, we allow these machines probabilistic choice. Initially, given an input string $\omega \in \{0,1\}^*$, a probabilistic P-RAM consists of a single probabilistic RAM initialized as defined in 2.3, with an additional operation: *fork* which allows the original RAM to create a new "clone" RAM sharing the same memory, with copies of the original RAM's registers with the same contents, with an identical finite state control, and initialized at some given state. Any new RAMs may also create new RAMs by the fork operations. All these RAMs operate synchronously with the original RAM. Furthermore, their probabilistic choices are assumed to be independent. RAMs are allowed to simultaneously read the same memory location. However, if two distinct RAMs simultaneously load into the same memory contents, then the entire computation of the P-RAM fails. If on a particular computation sequence the original RAM enters its accept state and there have been no such simultaneous memory load conflicts then this computation sequence is considered to be accepting. The probabilistic P-RAM accepts an input string $\omega \in \{0,1\}^*$ if with probability $> 1/2$ a random computation sequence is accepting. (See 2.2 for definitions of errors of acceptance and rejection.) The probabilistic P-RAM has *time bound* $T(n)$ (*space bound* $S(n)$, *integer bound* $I(n)$, *processor bound* $P(n)$) if on all inputs of length n and accepting computation sequences, the machine taken $\leq T(n)$ steps, (uses $\leq S(n)$ space, operates on integers $\leq I(n)$, uses $\leq P(n)$ processors, respectively). Note that space and time are charged in units depending on whether the machine is unit-cost or log-cost as defined in 2.3.

3. SOME FAST PROBABILISTIC PARALLEL ALGORITHMS

This section describes some time efficient algorithms for probabilistic P-RAMs which we easily derive by parallelizing known probabilistic sequential algorithms. (Section 4 gives a uniform method for parallelizing any probabilistic sequential algorithm.) All the algorithms described here are R-*probabilistic*: with rejection error $< 1/2$ (and no errors of acceptance) if the probabilistic trials are made twice.

THEOREM 3.1. *There are unit-cost* R-*probabilistic* P-RAMs *with time bound* $O(\log n)$ *and processor bound* $O(n^3 \log n)$, *which given a graph* G *with* n *vertices,*

(a) *can test if* G *has a path between two given vertices, and*

(b) *can also test if* G *is bipartite.*

Proof. [Aleliuneas, et al., 79] give for these problems R-probabilistic sequential algorithms which can be implemented on a probabilistic RAM in $O(1)$ space (using integers size $\leq n^2$ for representing edges) and $O(n^3)$ time. Our probabilistic parallel algorithms are derived immediately by applying Theorem 4.1. □

Note that the fastest known deterministic P-RAM algorithm for testing connectivity requires $O(\log n)^2$ time and $O(n^5)$ processors [Csanky, 76].

THEOREM 3.2. *A unit-cost* R-*probabilistic* P-RAM *with time bound* $O(\log n)^2$ *and processor bound* $O(n)$ *can test if a graph of* n *vertices has a perfect matching.*

Proof. Let $G = (V,E)$ be a simple graph with vertices $V = \{1,\ldots,n\}$. Lovasz, 80] gives a probabilistic sequential algorithm which chooses an $N = n^{O(1)}$ and constructs a symmetric $n \times n$ matrix $B = B_{ij}$ where for $1 \leq i,j \leq n$

(a) B_{ij} is a random element of $\{1,\ldots,N\}$ if $i < j$ and $(i,j) \in E$.

(b) $B_{ij} = -B_{ji}$ if $i > j$ and $(i,j) \notin E$.

(c) $B_{ij} = 0$ otherwise.

If the determinant of B is not 0 then G has a perfect matching. If the determinate of B is 0, then for N sufficiently large, G has a perfect matching with probability $< 1/2$. The parallel matrix inversion algorithm of [Csanky, 76] can be used to compute the determinant in time $O(\log n)^2$ and $O(n^5)$ on a P-RAM. □

THEOREM 3.3. *A unit-cost* R-*probabilistic* P-RAM *with* $O(n + (\log(nm))^2)$ *time bound and* $O(n+m)$ *processor bound can test if a polynomial* $f(x)$ *of degree* m *has a root in* $GF(p^n)$, *where* p *is a fixed prime.*

Proof. We parallelize the probabilistic algorithm of [Rabin, 80] (which generalized and proved validity for a previous algorithm of [Berlekamp, 70] for $GF(p)$). (This algorithm can be implemented on a unit-cost probabilistic sequential RAM in time $O(n^2m)$). First, compute $f_1(x) = GCD(f(x), x^{p^n-1} - 1)$. If $f_1(x) = 1$ then $f(x)$ has no roots over $GF(p^n)$. Otherwise, choose a random $\delta \in \{0,1,\ldots,p^n-1\}$ and compute $f_\delta(x) = GCD(f_1(x), (x+\delta)^{(p^n-1)/2})$. Let d_1, d_δ be the degrees of polynomials $f_1(x)$, $f_\delta(x)$ respectively. If $0 < d_\delta < d_1$ then $f(x)$ has a root in $GF(p^n)$ (in this case $f(x)$ has factor $f_\delta(x)$ if $2d \leq d_1$ and factor $f_1(x)/f_\delta(x)$ if $2d_\delta > d_1$), and otherwise $f(x)$ is irreducible in $GF(p^n)$ with probability $\geq 1/2$. The required polynomial GCD computations can be done by a unit-cost P-RAM $O(\log(nm))^2$ time and $O(n+m)$ processors by using the shuffle-exchange network of [Stone, 71] to compute the convolutions required for the polynomial GCD algorithm of [Aho, Hopcroft, and Ullman, 74]. The exponentiations can be computed in $O(n)$ parallel time by repeated exponentiation. □

(Note that the fastest known deterministic sequential algorithms [Adleman, 80] and [Adleman and Odlyzko, 81] for testing if a polynomial of degree n has a root over $GF(p^n)$ require time $O(\log n)^{\log(\log(\log n)))}$. These algorithms be speed-up by our Theorem 5 to $O(\log n)^{1/2 \log(\log(\log(n)))+1}$ parallel time on a deterministic P-RAM, but the resulting parallel algorithms remain very slow in comparison to those provided by Theorems 3.3.

THEOREM 3.4. *A unit-cost* R-*probabilistic* P-RAM *with time bound* $O(\log n)$ *and processor bound* $O(n^2/\log n)$ *given* $n \times n$ *integer matrices* A, B, C *can test* $A \cdot B \neq C$.

Proof. Choose a random column vector $x \in \{-1,1\}^n$ and test $A(Bx) \neq Cx$. This test can be done by a probabilistic P-RAM within time $O(\log n)$ and processor bound $O(n^2/\log n)$ by forming $n/\log n$ binary trees of processors, each of size $2n$ and depth $O(\log n)$, and pipelining the required dot products. [Freivalds, 79] shows that if $A \cdot B \neq C$ then $\text{Prob}\{A(Bx) = Cx\} \geqslant 1/2$. □

Note that the naive algorithm for testing $A \cdot B \neq C$ in time $O(\log n)$ on a deterministic P-RAM requires at least $n^3/\log n$ processors.

4. SIMULATION RESULTS BETWEEN PROBABILISTIC RAMs AND PROBABILISTIC P-RAMs

[Fortune and Wyllie, 78] and [Wyllie, 79] characterize the computational complexity of their deterministic P-RAMs in terms of the complexity of deterministic complexity classes. It is the aim of this section to do the same for our probabilistic P-RAMs. Our simulation methods are similar, except for the use of probabilistic choice to insure the probability of errors of acceptance and rejection are preserved.

4.1 Simulation of a Probabilistic RAM by a Probabilistic P-RAM

THEOREM 4.1. *Let* M *be a probabilistic* RAM *with constructible time bound* $T(n) \geq n$, *space bound* $S(n) \geq \log n$, *and integer bound* $I(n)$. *Then there is a probabilistic* P-RAM M' *such that* $M \approx M'$ *(see 2.2 for definition of the equivalence relation* $\approx$*); if* M *is unit-cost then* M' *has unit-cost time bound* $O(S(n) \log I(n) + \log T(n))$, *and processor bound* $O(I(n)^{S(n)} T(n))$; *if* M *is log-cost then* M' *has log-cost time bound* $O(S(n) + \log T(n))^2$ *and processor bound* $O(4^{S(n)} T(n))$.

(Note: Theorem 4.1 gives a speed-up for unit-cost RAMs only if $S(n) \log I(n) < T(n)$; Theorem 5.1 provides a uniform quadratic speed-up even if $S(n) = T(n)$.)

Proof. Fix some input string $\omega \in \Sigma^n$ and let $I_0(\omega)$ be the initial configuration of M. Let $\mathcal{I}$ be the set of configurations of M with space $S(n)$. Let $p = |\mathcal{I}|(T(n) + 1)$. Let each $I \in \mathcal{I}$ and each t, $0 \leq t \leq T(n)$ be encoded as a distinct integer $i = \langle I, t \rangle$, where $1 \leq i \leq p$. We can assume that the encoding and its decoding are computed in $O(\log p)$ steps on a P-RAM.

Our simulating probabilistic P-RAM M' will begin by a series of fork operations yielding RAMs $M_1,\ldots,M_p$. Each RAM M_i, $1 \leq i \leq p$, has a local register r_i and an associated global memory location NEXT_i which is initialized as follows: suppose $i = \langle I,t\rangle$ then if I has any immediate successor I', let M_i randomly choose some such I' and load $\langle I',t+1\rangle$ into NEXT_i and otherwise if I has no successors then let M_i load i into NEXT_i. After this initialization, each M_i, for $1 \leq i \leq p$, synchronously:

(1) loads the contents of NEXT_j into register r_i where j is the contents of NEXT_i, and

(2) then loads NEXT_i with the contents of r_i.

This is repeated $\lceil \log p \rceil$ times. We can assume $\langle I_0(\omega),0\rangle = 1$ and M_1 is the original RAM of M'. We let M_1 enter the accepting state (so M' accepts) if NEXT_1 ever contains integer $\langle I,t\rangle$ where I is an accepting configuration of M.

If M' accepts on a particular computation, then there must be a sequence of memory locations $\mathrm{NEXT}_{\langle I_0,0\rangle},\ldots,\mathrm{NEXT}_{\langle I_{t-1},t-1\rangle}$ is initialized to $\langle I_1,1\rangle,\ldots,\langle I_t,t\rangle$ where $I_0(\omega) = I_0$, $I_1,\ldots,I_t$ is an accepting computation sequence of M, and $t \leq T(n)$. Thus the memory essentially forms a path from $\mathrm{NEXT}_{\langle I_0,0\rangle}$ to $\mathrm{NEXT}_{\langle I_t,t\rangle}$ decreases by a factor of 1/2. Thus after $\lceil \log p \rceil$ iterations, $\mathrm{NEXT}_{\langle I_0,0\rangle}$ contains $\langle I_t,t\rangle$.

Suppose $I_0,I_1,\ldots$ is an execution sequence of M, derived from a particular sequence of probabilistic choices ρ. Suppose also that the RAMs of M' make a sequence of probabilistic choices ρ' such that $M_{\langle I_t,t\rangle}$ initially loads $\mathrm{NEXT}_{\langle I_t,t\rangle}$ with $\langle I_{t+1},t+1\rangle$ for $t = 0,1,\ldots,T(n)-1$. Then M errors on acceptance (rejection, respectively) of ω when making probabilistic choices ρ iff M' errors on acceptance (rejection, respectively) of ω when making probabilistic choices ρ'. Since ρ and ρ' are chosen randomly, it follows that $M \approx M'$. If M is unit-cost $|\mathcal{S}| \leq I(n)^{S(n)}$; so if M' is also considered to be unit-cost the time and space bound is $O(\log p) = O(S(n)\log I(n) + \log T(n))$ and the processor bound is $p = O(I(n)^{S(n)}T(n))$. If M is log-cost $|\mathcal{S}| \leq 2^{2\cdot S(n)} = 4^{S(n)}$; so if M' is also considered to be log-cost its time bound is $O(\log p)^2 = O(S(n) + \log T(n))^2$ and processor bound is $p = O(4^{S(n)}T(n))$. □

4.2 Simulation of a Probabilistic P-RAM by Probabilistic RAM

THEOREM 4.2. *Let* M *be a probabilistic* P-RAM *with time bound* $T(n)$, *space bound* $S(n)$, *and processor bound* $P(n)$. *Then there is a probabilistic* RAM M' *with space bound* $O(S(n) + P(n))$ *such that* $M \approx M'$. *Furthermore, if* M *is unit-cost then* M' *has unit-cost time bound* $O(T(n)P(n))$; *and if* M *is log-cost then* M' *has log-cost time bound* $O(T(n)\, P(n)\log P(n))$.

Proof. The simulating probabilistic RAM will have only 5 registers; the first register of M' will store an integer p giving the total number of RAMs currently being executed, and the second register of M' will store an integer designating the RAM currently being simulated; the other 3 registers of M' will be used for arithmetic operations and indirect addressing of memory locations. Suppose each RAM of M has r registers. The registers of the simulated RAMs of M will be stored in a special block of memory locations, which is increased by $r+1$ on every fork operation. The simulation of M' by M is straightforward; on each move of M, M' must simulate a move by each of the currently active RAMs of M. This requires $O(P(n))$ steps if M' is unit-cost, and $O(P(n)\log P(n))$

steps if M' is log-cost. By storing two copies of the memory of M, it is easy to detect simultaneous load conflicts. M' is allowed to enter its accepting state just when the original RAM of M enters its accepting state and there are no simultaneous load conflicts. Since the probabilistic choices taken by the individual probabilistic RAMs are assumed to be independent, and the simulating probabilistic RAM M' takes independent probabilistic choices, the probability of errors of acceptance and rejection of M and M' are identical, so $M \approx M'$. □

5. PARALLEL SPEED-UP OF PROBABILISTIC RAMs

THEOREM 5.1. *Let* M *be a probabilistic* RAM *with constructible time bound* $T(n) \geq n$ *and integer bound* $I(n)$. *Then there is a probabilistic* P-RAM M' *such that* $M \approx M'$ *and if* M *is unit-cost then* M' *has unit-cost time bound* $O(T(n)(\log T(n))\log(T(n)I(n)))^{1/2}$ *and if* M *is log-cost then* M' *has log-cost time bound* $O(T(n)^{1/2}\log T(n))$.

Proof. Let $\omega \in \{0,1\}^*$ be on input string of length n.

There is a constant $c \geq 1$ such that M has at most c choices for next moves at each step. Thus the choices can be represented by a sequence $\rho = \rho_0, \ldots, \rho_{T(n)-1}$ where $\rho_t \in \{1,\ldots,c\}$. The parallel simulation of M by M' begins by probabilistically choosing $\rho_0, \ldots, \rho_{T(n)-1}$ in $O(\log T(n))$ parallel time, and storing these choices in distinct memory locations.

The fundamental idea (previously used in [Hopcroft, Paul, and Valiant, 75] and [Dymond, 80] for speed-up of deterministic Turing machines) is to partition the T(n) steps into consecutive intervals of length L, $1 \leq L \leq T(n)$ to be determined below.

Let q be the number of states in the finite controle of M. Suppose in the following that M is unit-cost. Then M can read from and load into at most 3L registers and memory locations within a time interval Δ of length L. Furthermore, we can encode by a positive integer $\leq r = q(T(n)I(n))^{3L}$ the current state and the contents and addresses of the registers and memory locations read from (or loaded into) during Δ. (If M is log-cost, M can read from and load into at most 3L bits of registers and memory locations with a time interval Δ of length L. Thus we can encode this by a positive integer $\leq r$, where $r = q(T(n)4)^{3L}$ in the case M is log-cost.)

Let $H = \lceil T(n)/L \rceil - 2$. For each $t = 0, L, 2L, \ldots, HL$ the simulating M' constructs in global memory a table $\mathrm{PREDICT}_t$ which given a positive integer $i \leq r$ encoding a possible state of M and contents and addresses of all registers and memory locations to be read during time interval $\Delta_t = \{t, t+1, \ldots, t+L-1\}$ $\mathrm{PREDICT}_t(i)$ is a positive integer $\leq r$ encoding the contents and addresses of all registers and memory locations to L loaded into during Δ_t using the predetermined choice sequence $\rho_t, \rho_{t+1}, \ldots, \rho_{t+L-1}$. However, let $\mathrm{PREDICT}_t(i) = 0$ if this choice sequence requires reading a register or memory location whose contents are not defined by i, or if the contents of a register or memory location are provided by i but are not read from. These tables can be constructed in parallel by M' in time $O(L + \log r)$.

$T(n)$ distinguished global memory locations of M' are used to store the contents of the memory of M. Also, a special register is used to store the state of the finite control of M. These are initialized as in the initial configuration of M. The simulation of M by M' will then proceed sequentially in H phases, each corresponding to a time interval Δ_t, for $t = 0, L, 2L, \ldots, HL$.

Suppose at the start of the phase corresponding to interval Δ_t, M' is currently storing (as described above) the configuration I_t of M, where $I_0, I_1, \ldots, I_t$ is the sequence of configurations of M induced from $I_0 = I_0(\omega)$ by the choice sequence $\rho_0, \rho_1, \ldots, \rho_{t-1}$ chosen by M' at the start of the simulation. Then there is a unique sequence of configurations $I_t, I_{t+1}, \ldots, I_{t+L}$ induced by the predetermined choice sequence $\rho_t, \rho_{t+1}, \ldots, \rho_{t+L-1}$. Hence there is a unique i_t, $1 \leq i \leq r$, such that $\mathrm{PREDICT}_t(i_t) \neq 0$ and i_t encodes contents of registers and memory locations consistent with I_t. $\mathrm{PREDICT}_t(i_t)$ is encoded and is used to update the memory of M' to store the configuration I_{t+L}. After the phase associated with time interval Δ_{HL}, M' simulates M step by step sequentially for $t=(H+1)L,(H+1)L+1,\ldots,T(n)$. Let the original RAM of M' enter the accepting state if the simulated M does. Since the choice sequence $\rho_0,\ldots,\rho_{T(n)-1}$ is chosen randomly by M', it induces a random computation sequence of M from $I_0(\omega)$, so $M \approx M'$.

In the case M is unit-cost, we let M' be unit-cost. The unit-cost time for initialization and computation of the PREDICT tables is $O(L+\log r) = O(L \log(T(n)I(n)))$. The unit-cost time for each phase is $O(\log\log r) = O(\log(L \log(T(n)I(n))))$ since encoding and decoding of elements of the PREDICT tables is done in prallel. There are $< T(n)/L$ phases. Thus the total unit-cost time is

$$O(L \log(T(n)I(n))) + (T(n)/L)O(\log(L \log(T(n)I(n)))) + L$$
$$= O(T(n)(\log T(n))\log(T(n)I(n)))^{1/2},$$

for

$$L = (T(n)(\log T(n))/\log(T(n)I(n)))^{1/2}.$$

In the case M is log-cost, we similarly let M' be log-cost. To allow for $O(\log\log r)$ parallel log-cost time access of the PREDICT tables, the $\log r$ bits of each element of a PREDECT table must be stored in distict contiguous memory locations, instead of a single memory location. The log-cost time for initialization and computation of the PREDICT tables is $O(L+\log r) + O(L \log T(n))$. The log-cost time for each phase is $O(\log\log r) = O(\log(L \log T(n))))$. Thus the total log-cost time is

$$O(L \log(T(n))) + (T(n)/L)\log(L \log T(n))) + L = O(T(n)^{1/2}\log T(n))$$

for

$$L = T(n)^{1/2}. \qquad \square$$

6. ELIMINATION OF PROBABILISTIC CHOICE IN PARALLEL COMPUTATIONS

Let N be a (uniform) probabilistic P-RAM with time bound $T(n)$ and processor bound $P(n)$. Let $Z(n)$ be the maximum number of probabilistic choices made by all the RAMs of N on any input of length n. (Note that $Z(n) \leq T(n)P(n)$). Let $\epsilon_A(n), \epsilon_R(n)$ be the acceptance and rejection error functions for N, and let $\epsilon(n) = \max(\epsilon_A(n), \epsilon_R(n))$. Also, let $\lambda(n) = (1+2n)/\log(1/(4\epsilon(n)(1-\epsilon(n))))$. We assume $\epsilon(n) < 1/2$ so $\lambda(n)$ is finite.

The following theorem states that we can eliminate the probabilistic choice in N by introducing *nonuniformity with advice bound* $\lambda(n)$: i.e., we allow the nonuniform P-RAM to have in the initial configuration for each input length $n \geq 0$, a distinguished sequence of $\lambda(n)$ memory locations each initialized to either 0 or 1 and fixed for all inputs of length n.

THEOREM 6. *For any* $\tau(n)$, $1 \leq \tau(n) \leq \lambda(n)$, *there is a deterministic nonuniform* P-RAM $\hat{N}$ *which accepts* $L(N)$ *with time bound* $O(T(n)\tau(n) + \log(\lambda(n)/\tau(n)))$, *processor bound* $O(P(n)\lambda(n)/\tau(n))$, *and advice bound* $O(\lambda(n)Z(n))$.

Note: Thus to eliminate probabilistic choice we have a trade-off between an increase in time bounds and an increase in processor bounds. However, if $\epsilon(n)$ decreases exponentially, then neither the time bound nor the processor bound are asymptotically increased.

Theorem 6 will be proved as follows: first we show that we can eliminate probabilistic choice from N if $\epsilon(n)$ is sufficiently small; then we show how to make $\epsilon(n)$ sufficiently small.

We can assume a constant $c \geq 1$ such that N has $\leq c^{P(n)}$ choices of moves next from any configuration. Fix some input length $n \geq 0$. A parallel *choice sequence* ρ is of the form $\rho_0,\rho_1,\dots,\rho_{T(n)-1}$ where $\rho_i \in \{1,\dots,c^{P(n)}\}$ for $i=0,1,\dots,T(n)-1$. Let $R_{T(n)}$ be all choice sequences of length $T(n)$. Given an input $\omega \in \{0,1\}^n$, a choice sequence in $S_{T(n)}$ induces a computation sequence of N. Let $R_{T(n)}(\omega) = \{\rho \in R_{T(n)} \mid (\omega \in L(N)$ and N has an accepting computation sequence on input ω and choice sequence $\rho)$ or $(\omega \notin L(N)$ and N has a nonaccepting computation sequence on input ω and choice sequence $\rho)\}$.

LEMMA 6.1. *If* $\varepsilon(n) < 2^{-n}$, *then there is a deterministic nonuniform P-RAM* $\hat{N}$ *which accepts* $L(N)$ *with time bound* $O(T(n))$, *processor bound* $P(n)$ *and advice bound* $O(Z(n))$.

Proof. It suffices to show (*):

(*) if $\varepsilon(n) < 2^{-n}$ then there exists some choice sequence $\rho^* \in R_{T(n)}$ such that for all $\omega \in \{0,1\}^n$, $\rho^* \in R_{T(n)}(\omega)$.

Our proof is by contradiction (and thus is not constructive). For each $\rho \in R_{T(n)}$ let $f(\rho) = |\{\omega \in \{0,1\}^n \mid \rho \in R_{T(n)}(\omega)\}|$ and let $r = |R_{T(n)}|$. Suppose (*) does not hold, so $2^n > f(\rho)$ for all $\rho \in R_{T(n)}$. Hence

$$\begin{aligned} 2^n &> \frac{1}{r} \sum_{\rho \in R_{T(n)}} f(\rho) \\ &= \frac{1}{r}\,(r/\varepsilon(n)) \\ &= 1/\varepsilon(n) \\ &> 2^n, \text{ a contradiction} \end{aligned}$$

□

LEMMA 6.2. *For any* $\tau(n)$, $1 \leq \tau(n) \leq \lambda(n)$, *there is a probabilistic P-RAM* N' *which accepts* $L(N)$ *with acceptance and rejection errors* $\varepsilon'_A(n)$, $\varepsilon'_R(n)$ *where* $\max(\varepsilon'_A(n),\varepsilon'_R(n)) < 2^{-n}$, *and time bound* $O(T(n)\tau(n) + \log(\lambda(n)/\tau(n)))$ *and processor bound* $O(T(n)\lambda(n)/\tau(n))$.

Proof. Let $\omega \in \{0,1\}^n$ be the input string, for some $n \geq 0$. Our probabilistic P-RAM M' will simulate M on input ω a total of $\lambda(n)$ times; these simulations will be done by $\lceil \lambda(n)/\tau(n) \rceil$ groups of $P(n)$ probabilistic RAMs, with each group simulating M $\tau(n)$ times. M' is allowed to enter an accepting configuration only if M enters an accepting configuration on at least $\lambda(n)/2$ of the $\lambda(n)$ trials. (This technique of determining the consensus of a series of trials is due to [Bennett and Gill, 81].) The count of successful trials can be computed in $\log(\lambda(n)/\tau(n))$ parallel time. The acceptance error of M' is

$$\begin{aligned} \varepsilon'_A(n) &= \sum_{i=\lambda(n)/2}^{\lambda(n)} \binom{\lambda(n)}{i} \varepsilon(n)^i (1-\varepsilon(n))^{\lambda(n)-i} \\ &\leq (4\varepsilon(n)(1-\varepsilon(n)))^{\lambda(n)/2} \quad \text{by bounds of [Chernoff, 52] also given in [Feller, 57]} \\ &< 2^{-n} \quad \text{for given } \lambda(n) > 2n/\log(1/(4\varepsilon(n)(1-\varepsilon(n)))). \end{aligned}$$

Also we can similarly show the error of rejection $\varepsilon'_R(n) < 2^{-n}$. Hence $\max(\varepsilon'_A(n), \varepsilon'_R(n)) < 2^{-n}$ as claimed. □

Theorem 6 follows immediately by applying to Lemma 1 the probabilistic P-RAM M' derived by Lemma 6.2.

By applying Theorem 6 to Theorems 3.1-3, we have:

COROLLARY 6.1. *There exists unit-cost nonuniform deterministic* P-RAMs *with time bound* $O(\log n)$, *processor and advice bound* $O(n^4 \log n)$, *which given a graph* G *with* n *vertices, can test* (a) *whether* G *has a path between two given vertices and can also test* (b) *whether* G *is not bipartite.*

COROLLARY 6.2. *There exists a unit-cost nonuniform deterministic* P-RAM *with time bound* $O(\log n)^2$, *processor and advice bound* $n^{O(1)}$ *which can test if a graph of* n *vertices has a perfect matching.*

COROLLARY 6.3. *There exists unit-cost nonuniform deterministic* P-RAMs *with time bound* $O(n)$ *, processor and advice bound* $O(n^2)$ *which can test:*

> *given a polynomial of degree* $O(n)$, *does it have a root in* $GF(p^n)$*?*

7. CONCLUSION

This paper has primarily considered the power of probabilistic choice for parallel RAMs. Theorems 3.2-5 also hold for fixed connection parallel networks with probabilistic processors. Theorems 4.1 and 4.2 can be extended to similar simulation results for other probabilistic parallel machines, such as the hardware modification machines (HMMs) of [Cook, 80] augmented with probabilistic choice (see [Reif, 81]). Also Theorem 4 generalizes to other probabilistic parallel machines such as HMMs and circuits with probabilistic choice.

ACKNOWLEDGMENTS

The author was informed by Larry Russo of the consensus technique previously used by [Bennet and Gill, 80] for decreasing errors of probabilistic choice. Paul Spirakis gave helpful comments on a reading a preliminary draft of this paper. Renate D'Arcangelo is sincerely thanked for an excellent typing of this paper.

REFERENCES

Adleman, L., "Two theorems on random polynomial time," Proceedings of the 19th IEEE Symposium on the Foundations of Computer Science, Ann Arbor, MI, 1978, pp. 75-83.

Adleman, L., "On distinguishing prime numbers from composite numbers," Annual Symposium of Foundations of Computer Science, 1980.

Adleman, L. and K. Manders, "Reducibility, randomness and intractability," Proceedings of the 9th ACM Symposium on the Theory of Computing, 1977, pp. 151-153.

Adleman, L., Manders, K., and G. Miller, "On taking roots in finite fields," IEEE Symposium on the Foundations of Computer Science, 1977, pp. 175-178.

Adleman, L. and Odlyzko, A., "Irreducibility testing and factorization of polynomials," 22nd Annual Symposium on Foundations of Computer Science, 1981, pp. 409-420.

Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Pub. Comp., Reading, Mass. ,1974, pp.303-310.

Aleliunas, R., R.M. Karp, R.H. Lipton, L. Lovasz and C. Rackoff, "Random walks, universal traversal sequences, and complexity of maze problems," Proc. 20th Annual Symposium on Foundations of Computer Science, 1979, pp. 218-223.

Angluin, D., "Local and global properties in networks of processors," 12th Annual Symposium on Theory of Computing, Los Angeles, California, April 1980, pp. 82-93.

Barzdin, A.M., "On computability by probabilistic machines," Dokl. Akad. Nauk SSSR, 189 (1969), pp. 699-702, = Soviet Math. Dokl., 10 (1969), pp. 1464-1467.

Bennett, C.H. and Gill, J., "Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1," *SIAM J. Comput.*, vol. 10, No. 1 (Feb. 1981), pp. 96-113.

Berlekamp, E.R., "Factoring polynomials over large finite fields," *Math. Comp.* 24, (1970), pp. 713-735.

Chandra, A.K., D.C. Kozen and L.J. Stockmeyer, "Alternation," *J. ACM*, 1981.

Cook, S.A., "Towards a complexity theory of synchronous parallel computation," Presented at Internationales Symposium über Logik und Algorithmik zu Ehren von Professor Horst Specker, Zürich, Switzerland, February 1980.

Csanky, L., "Fast parallel matrix inversion algorithms," *SIAM J. Comput.* 5, (1976), pp. 618-623.

Chernoff, H., "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. of Math. Stat.*, vol. 23, (1952), pp. 493-507.

Dymond, P.W., "Speedup of multi-tape Turing machines by synchronous parallel machines," Technical Report, Dept. of EE and Computer Science, Univ. of California, San Diego, California.

Dymond P., and S.A. Cook, "Hardware complexity and parallel computation," IEEE FOCS Conference, 1980.

Feller, W., *An Introduction to Probability Theory and its Applications*, John Wiley, New York, 1957.

Freivalds, R., "Fast Probabilistic Algorithms," 8th MFCS, 1979.

Fortune, S. and J. Wyllie, "Parallelism in random access machines," In Proc. of the 10th ACM Symposium on Theory of Computation, 1978, pp. 114-118.

Francez, N. and Rodeh, "A distributed data type implemented by a probabilistic communication scheme," 21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, Oct. 1980, pp. 373-379.

Gill, J., "Complexity of probabilistic Turing machines," *SIAM J. of Computing*, 6(4), 675-695 (1977).

Goldschlager, L., "A unified appraoch to models of synchronous parallel machines," In Proc. 10th Annual ACM Symposium on the Theory of Computing, San Diego, California, 89-94 (1978).

Hirschberg, D.C., "Parallel algorithms for the transitive closure and the connected components problems," In Proc. 8th Annual ACM Symposium on the Theory of Computing, 55-57 (1976).

Hopcroft, J.E., and Karp, R.M., "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comp.*, vol. 2, No. 4, (Dec. 1973), pp. 225-231.

Hopcroft, J.E., W. Paul, and L. Valiant, "On time versus space and related problems," IEEE 16 SWAT, 1975.

Iberra, O.H., and S. Moran, "Probabilistic algorithms for deciding equivalence of straight-line programs," Computer Science Dept., University Minnesota, Tech. Report 80-12. (March, 1980).

Lehman, D. and M. Rabin, "On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers' problem," to appear in 8th ACM Symposium on Principles of Program Languages, Jan. 1981.

Lovasz, L., "On determinants, matchings, and random algorithms," to appear, 1980.

Rabin, M.O., "Probabilistic algorithms," *Algorithms and Complexity, New Directions and Recent Results*, edited by J. Traub, Academic Press, 1974.

Rabin, M.O., "Probabilistic algorithms in finite fields," *SIAM J. Comp.* 9, No. 2 (May 1980), pp. 273-280.

Rabin, M.O., "N-process synchronization by a $4 \log_2 N$-valued shared variable," 21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, Oct. 1980, pp. 407-410.

Reif, J.H., "Symmetric complementation," Technical Report TR-07-81, Aiken Computation Laboratory, Harvard University, Oct. 1981.

Reif, J.H. and P. Spirakis, "Distributed algorithms for synchronizing interprocess communication within real time," 13th Annual ACM Symposium on the Theory of Computing, Milwaukee, Wisconsin, 1981.

Reif, J.H. and P. Spirakis, "Unbounded speed variability in distributed communication systems," 9th ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico, Jan. 1982.

Reischuk, R., "A fast probabilistic parallel sorting algorithm," 22nd Annual Symposium on Foundations of Computer Science, Nashville, Tenn.,Oct. 1981.

Savitch, W., and M. Stimson, "Time random access machines with parallel processing," *J. ACM* 26, 108-118 (1979).

Schwartz, J.T., "Fast probabilistic algorithms for verification of polynomial identities," *JACM* 27, (4), pp. 701-717 (Oct. 1980).

Simon, J., "On some central problems in computational complexity," TR75-224, Dept. Comp. Science, Cornell Univ., Ithaca, NY, 1975.

Solovay, R. and Strassen, V., "A fast Monte-Carlo test for primality," *SIAM J. of Computing* 5, (1), pp. 84-85 (1977).

Stone, H.S., "Parallel processing with the perfect shuffle," *Trans. on Computers*, C-20, (2), pp. 153-161 (Feb. 1971).

Valiant, L.G., "A scheme for fast parallel communication," Technical Report, Computer Science Dept., Edinburg Univ. Edinburg, Scotland, July 1980.

Wyllie, J.C., "The complexity of parallel computations," Ph.D. Thesis and TR-79-387, Dept. of Computer Science, Cornell University, 1979.

Yemini, Y., "Some theoretical aspects of position location problems," Proc. of the 20th Annual Symposium on Foundations of Computer Science, pp. 1-8 (1979).

Zippel, R., "Probabilistic algorithms for sparse polynomials," EUROSAM Proceeding, 1979.